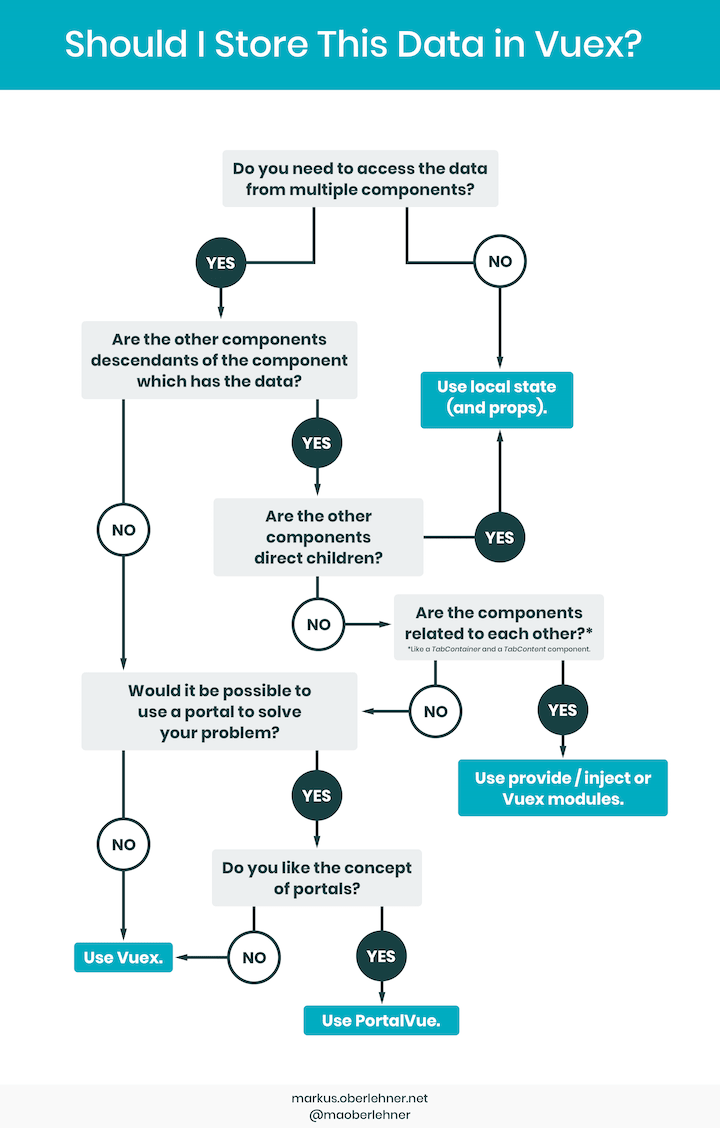
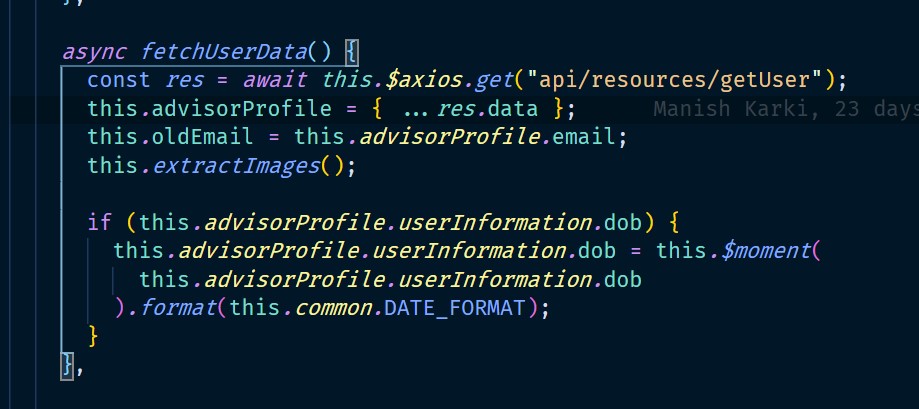
* 1. Always, always write meaningful names, create short and sweet names for variables and files as much as you can but if not possible, it could be a little longer but use correct spellings and consistent naming pattern.
  2. User lowercase for folder names and upper camel case for file names.
  3. Always initialize with null instead of "". If you need a string make it a string when you need to. And above all keep it consistent, there are "" in some places and null in others right now. Manish's variable is mostly "", Susan's and Asim's is mostly null. Need to make it consistent.
  4. Don’t leave unused and unnecessary comments and commented code. Delete it. Use comments for very complex logic to give brief description of the code before the code block, but ultimately having to write no comments and keeping the code itself readable in almost English is much better although it might not always be possible in complex use cases or data manipulation functions.  
     If you have to comment a code block, give it a context, explaining why it is commented
  5. Use Vuex only if needed,  
     let’s assume user opens profile page,  
     there you have to fetch the profile data on mounted,  
     then there is no benefit of saving the user data on Vuex state,  
     the data still needs to be fetched on mounted because the data might have been changed on some other session use might have on some other computer and on each page refresh it has to be fetched again.  
     Vuex is more useful on shared state, when same data decides the behavior of different independent components which are not parent children related.



* 1. Try to use modularization in Vuex if possible.
  2. When you need is just a data object because api hasn’t been implemented for the said data to be fetched from the database, use JavaScript file which contains the data and import it like commons.js, you could create data.js or something like that.
  3. Route paths would be better to be like /meetings/upcoming instead of /upcomingmeetings. Kind of like, subject/type/id, subjects like meetings, prospect, advisors, etc. first, and then type of the subject like upcoming, previous, etc. and then finally if you need to be specific or pass some id or slug or some other data to the backend to be specific then pass those. This will match most restful api, for the rest you need to adapt but try to keep the pattern consistent if possible.

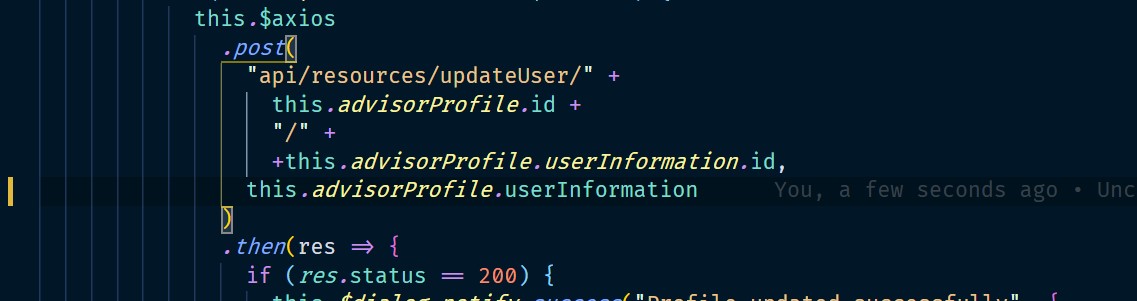
Others:

This is the only place that has this problem so far i have seen,



Here, the response data is directly spread into the advisor profile variable

This is the advisor profile in that page, you seem to have arranged the data structure according to the response structure. So far so good.



Now, here during saving the profile data in the same page, you were just submitting just the user information back to the server, which did not include email and consequently email could not be updated.



Fixed it like so.

The takeaway from this is that whatever the backend returns, maintain your own data structure, and distribute the data as required, this issue wouldn’t have been there if email was inside user information object in the first place. This is not that big of an issue and haven’t seen others like it so far so just letting you know.